

© 2012 Austin Pierce

USING INFORMATION THEORETIC MEASURES TO EVALUATE
SUPPORT VECTOR MACHINE KERNELS

BY

AUSTIN PIERCE

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Adviser:

Professor Richard Blahut

ABSTRACT

A new method is proposed that exploits the underlying information theoretic structure in the input data to evaluate the ability of a kernel to successfully separate a class in some feature space. This method is built on the fundamental idea that kernel density estimation in some input space is equivalent to an inner product on some Hilbert space. Estimators of Rényi's generalized form of information theoretic measurements reduce to a form that gives an elegant characterization of the geometric properties of the kernel in the feature space. It is shown how these estimators can be used to evaluate the kernel of a support vector machine.

*But I have promises to keep,
And miles to go before I sleep,
And miles to go before I sleep.*
- *Robert Frost*

ACKNOWLEDGMENTS

I think this is as close as I will ever get to a Grammy speech. First, I want to thank my parents for their never ending support, love, and guidance. Without them I would not be half the person I am today. This paper is dedicated to them. To my sister for constantly reminding me that hard work also requires hard play. You only live once, YOLO. “I don’t care how poor a man is; if he has family, he’s rich.”

I would like to thank Science Applications International Corporation (SAIC) for giving me the opportunity to work four summers alongside some of the smartest people I have ever met. The experience I have gained working there is invaluable and completely transformed how I approach engineering. This thesis would not have been possible without the conversations I had with Mike Tinston.

I am thankful to Prof. Richard Blahut for accepting me into his research group and for providing insightful weekly meetings.

TABLE OF CONTENTS

CHAPTER 1	INTRODUCTION	1
1.1	Machine Learning	1
1.2	Learning Model	4
1.3	Motivation	6
CHAPTER 2	INFORMATION THEORY	8
2.1	Shannon Entropy	8
2.2	Rényi Entropy	11
2.3	Rényi Entropy vs. Shannon Entropy	13
2.4	Quadratic Rényi Entropy	14
2.5	Cauchy-Schwarz Divergence	17
CHAPTER 3	REPRODUCING KERNEL HILBERT SPACES	19
3.1	RKHS Definitions	19
3.2	Density Estimation and RKHS	21
CHAPTER 4	SUPPORT VECTOR MACHINES	24
4.1	Vapnik-Chervonenkis Theory	24
4.2	Support Vector Machine	25
4.3	Kernel Trick	28
4.4	Evaluating Kernels Using Information Theory	29
CHAPTER 5	CONCLUSION	32
REFERENCES	33

CHAPTER 1

INTRODUCTION

The decision-making process behind identifying or classifying objects is often associated with the recognition of patterns. For example, determining the context and language of handwritten characters, using facial structure to detect individuals in photos, and deciding whether or not to buy or sell a stock based on current and past market trends. Recognizing patterns, collecting raw data, or making an action based on specific features of the input is a complex process that the human mind manages on a daily basis. For this reason, the creation of machines capable of learning patterns has been a popular research topic both from a technical and philosophical perspective. Technological advances in computer processing have allowed machines to demonstrate a significant ability to learn and recognize patterns. However, in order to demonstrate these tasks using classical programming techniques, there must be some foundation for the mathematical model of the problem. The methodology and theory behind the mathematical approach has formed the foundation for machine learning.

1.1 Machine Learning

Artificial intelligence is the term coined to describe the capability of machines to mimic and simulate intelligent human behavior. Machine learning has evolved as a branch of artificial intelligence that is concerned with understanding how to give machines the ability to learn and analyze data to make intelligent decisions. When computers are applied to solve a problem, they all follow the same practical architecture. The input data is fed through some system and mapped to an output. It is up to the system designer to develop a method for implementing the relationship between the input/output pairs. However, in some cases the relationship is not explicit, constantly chang-

ing, or the relationship is so complex that the computation is too expensive. When the problem or relationship is too complicated, the system designer cannot always implement a method for computing the output from the input data using classical programming techniques. The alternative method is to develop a system capable of learning or recognizing the pattern between the input and output based on examples, or given some specific criterion.

Learning in this context is achieved using mathematical principles. The framework is built on inductive inference; that is, observing samples that may or may not completely represent the input and predicting the output [1]. Generally, the input and output are related through some kind of dependency. When the dependency exists, it is referred to as the environment operator. The goal of the machine is to learn, and according to some cost, estimate the environment operator. The estimate is often referred to as the decision operator or classifier. Because the environment is not always completely understood, the decision operator is often chosen from a particular set or class of hypothesis operators. The learning process is simply choosing the appropriate operator, or classifier, from the class of hypothesis operators that best represents the environment operator.

Any method that attempts to describe the environment operator according to some prior knowledge employs learning. While learning follows a general model, it is the specific method used that determines the unknown parameters of the model and the structure of the classifier. Ideally, the classifier will generalize well to new patterns and data. How well the classifier performs on new data is referred to as generalization, and is the property that we seek to optimize. This framework results in two general forms of learning: supervised learning and unsupervised learning.

1.1.1 Supervised Learning

Supervised learning is primarily concerned with the identification of functional dependencies between the input x_i and output z_i . The environment operator is assumed to be some kind of functional mapping that is estimated from pairs of samples (x_i, z_i) called the training data. Figure 1.1 shows the basic framework for a supervised learning system. The data x_i is fed through the estimated environment operator to produce an output y_i . The output y_i

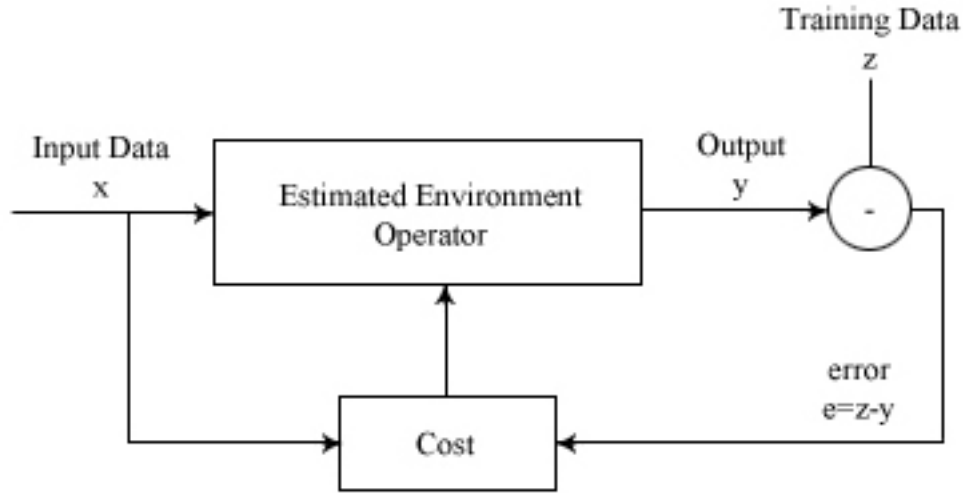


Figure 1.1: A basic supervised learning system.

is then compared to z_i to produce an error signal. The error signal is used to systematically update the environment according to some cost criterion. Over time the system will adapt the estimated environment operator so that the data x will produce a y that will approximate z .

A classifier that accurately fits the training data is said to be *consistent*. Generating consistent classifiers raises important issues. In cases where the output is noisy, there is no guarantee that a functional mapping exists. So, the classifier has to be implemented optimizing some best-fit criteria. If the classifier is designed to be too complex, it is possible to overfit the data. When this occurs, the classifier may correctly separate the training samples but will fail to generalize well to new data. In supervised learning, the design is concerned with optimizing the tradeoff between accuracy and reliability.

1.1.2 Unsupervised Learning

The goal of unsupervised learning is to capture regularities and statistical dependencies between the input x_i and output y_i . An unsupervised system follows the same framework as the supervised system in Figure 1.1 only x is available but z is not. The environment operator is assumed to function according to a set of specifications. The function is typically controlled by a cost. The classifier is designed to use the cost to form clusters or groupings of

the input. The output y_i from the classifier is entirely dependent on the cost that is implemented to replicate the environment operator. In unsupervised learning the design is primarily concerned with optimizing the cost to create a classifier that appropriately represents the regularities of the input data.

1.2 Learning Model

Now that we have introduced the foundation for machine learning, we can start to develop the systematic model for implementation on a computer [2]. Figure 1.2 shows the typical layout and components of a machine learning system. Understanding the importance of each component will clarify the problems with designing such a system and establish the motivation behind this research.

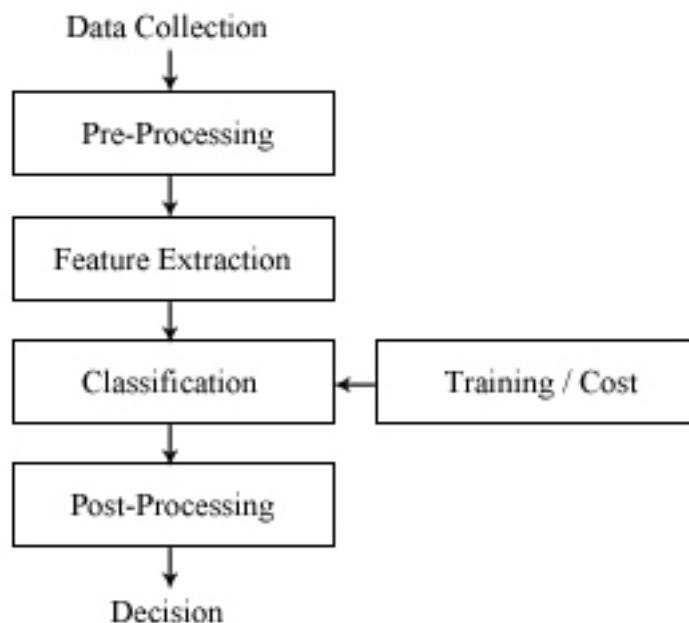


Figure 1.2: The block diagram for a typical machine learning system.

1.2.1 Data Collection

Data is typically collected using some kind of transducer, such as a microphone or sensor. The limitations and characteristics of the transducers, such as the signal-to-noise ratio or distortion, are often part of the problem. These limitations feed through the system and can affect each component if they are not considered in the design process. The amount of data that is collected and processed can significantly constrain other components, especially if the system is expected to work in real-time.

1.2.2 Pre-Processing

The data that is collected by the transducer is typically referred to as raw data. The raw data is often noisy, disorganized, or not in a format capable of being processed by the rest of the system. The purpose of the pre-processor is to prepare the raw data for feature extraction. Often this requires some kind of filtering or grouping operation. Ideally the pre-processor will isolate specific idiosyncrasies in the data.

1.2.3 Feature Extraction

The purpose of the feature extractor is to find distinguishing properties in the data that can be used to characterize an object or class. These distinguishing features are what the classifier will use to isolate the input and predict the output. When choosing these features, it is important that they are invariant to irrelevant transformations. For example, the data could get scaled or rotated in the pre-processing stage. If the distinguishable feature is affected by one of those transformations, then the feature will make it difficult for the classifier to reliably separate the input. The feature extractor is entirely domain-dependent. An extractor that successfully characterizes one problem will not necessarily work well for another.

1.2.4 Classification

The classifier is the most important component of the system and the most complicated. Because perfect classification is often impossible, the classi-

fier is built on a probabilistic framework. The general task is to use the distinguishing features to assign the input to a group or category with the smallest probability of error. The design is application dependent and follows one of the two general forms of learning: supervised or unsupervised. In the supervised case the classifier is fed training data to estimate a functional relationship between the input and output. In the unsupervised case the classifier is given a cost or criterion to group the output based on statistical dependencies of the input. The difficulty behind designing the classifier often depends on the size of the feature space and the difference between the distinguishing features in different categories. If the distinguishing features are drastically different, then the classifier dramatically simplifies. However, if the features are similar or if they are very noisy, the classifier can become much more complex.

1.2.5 Post-Processing

While the classifier analyzes the distinguishing features and classifies the data, it is up to the post-processor to decide the final action. Typically the post-processor is used to tweak the output of the classifier and improve system performance. This modification can be as simple as filtering the data to reduce output noise, or as complicated as regularizing the output to meet some specification.

1.3 Motivation

While it is not necessarily obvious, it should be relatively straightforward to notice that the problems and difficulties in designing each component can be manipulated in the classifier. For example, the amount of data collected during the collection stage is irrelevant if the classifier is capable of handling large amounts of data. Likewise, the variation in the distinguishable features can be changed if the classifier takes advantage of some kind of transformation. For this reason, this thesis will focus entirely on the design of the classification component of the learning model. Recall that the classifier is designed according to one of the two general forms of learning: supervised or unsupervised. Supervised learning requires the estimation of some func-

tion to mimic the environment operator, and unsupervised learning attempts to find the statistical dependency in the input data. So the classifier is inherently built on a probabilistic framework. While a Bayesian or maximum likelihood approach is reasonable and well understood, a new question is proposed: Are there other well-established probabilistic theories that can be used to take advantage of the statistical properties of the input? If so, what advantages does it provide? As it turns out, the answer lies in the field of information theory. Specifically, in Rényi's generalization of information theory. We will show that it is possible to evaluate how well a kernel function separates data in some feature space using information theoretic estimations on the input data. This method is built on the fundamental idea that kernel density estimation in some input space is equivalent to an inner product on some Hilbert space. Estimators of Rényi's generalized form of information theoretic measurements reduce to a form that gives an elegant characterization of the geometric properties of the kernel in the feature space. We apply this kernel evaluation to the design of a Support Vector Machine.

This thesis is organized as follows. In Chapter 2 we introduce the idea of α th-order information theoretic measures and their estimators. We develop the foundation for reproducing kernel Hilbert spaces in Chapter 3 to help form an important geometric understanding of these measures. In Chapter 4 we introduce the Support Vector Machine and propose a method for evaluating the feature spaces of kernels using information theory. We make concluding remarks and offer further research ideas in Chapter 5.

CHAPTER 2

INFORMATION THEORY

Information theory was originally conceptualized by Shannon to help establish the fundamentals behind data compression and transmission rates in communication systems [3]. For this reason, information theory is often considered to be a branch of communication theory. However, the probabilistic framework has laid the foundation for contributions in other areas of math and the sciences [4]. We introduce information theoretic measurements here as a probabilistic reasoning method for analyzing the statistical properties of data. Keep in mind that the motivation behind doing this is to exploit these statistical descriptors as tools when designing a classifier. We will start by introducing the original ideas developed by Shannon and explain the purpose of entropy, divergence, and mutual information. This will lead to a generalized form of information theory developed by Rényi. We will show that the estimator for Rényi entropy reduces to a form that is easier to compute than Shannon entropy.

2.1 Shannon Entropy

In 1928 Hartley introduced the first quantitative measure of information. He defined this information as the number of choices in a finite set of possible symbols S [5]. According to Hartley, the amount of information in a sequence of N symbols could be quantified as $I = \log_b S^N$. If the logarithm is base ten, then we refer to the unit as a Hartley. In the digital age we prefer to think in terms of binary symbols $[0,1]$, so the logarithm is taken to the base two and is referred to as the bit.

It was Shannon who went a step further and introduced the idea that the information content was dependent on the probabilities of the symbols. Let us define a random variable X with a finite set of possible outcomes

$\mathcal{X} = \{x_1, \dots, x_N\}$ having a probability mass function $p(x) = \Pr\{X = x\}$, $x \in \mathcal{X}$. Shannon first showed that Hartley's information was only accurate if nothing was known about the distribution of the symbols; that is, we assume that the symbols occur with equal probability $p(x) = \frac{1}{|\mathcal{X}|}$. Thus, in order to fully characterize the amount of information for each element x_k in \mathcal{X} , the information should be

$$I(x_k) = \log \frac{1}{p(x_k)} = -\log_2 p(x_k). \quad (2.1)$$

The amount of information in a specific event is due to the inverse dependence on probability. If $N = 1$, then $p(x_1) = 1$ and the information content $I(x_1)$ is zero. This means we have perfect knowledge of the event since we know there is only one possible event outcome. Likewise, if N is large and $p(x)$ is small, then the information content is high because the event is unexpected. To describe the information or uncertainty over the set of outcomes \mathcal{X} , Shannon defined entropy as the average of the information measure over each event, or

$$H(X) = \sum_{x \in \mathcal{X}} p(x) I(x) = - \sum_{x \in \mathcal{X}} p(x) \log_2 p(x) = -E[\log_2 p(x)]. \quad (2.2)$$

Shannon's entropy provides a single scalar quantity that describes the uncertainty underlying the probability mass function (PMF). In other words, $H(X)$ depends on the shape of the distribution. It is not the only scalar quantity that describes the PMF; for example, the mean and variance are also widely used. However, entropy provides a deeper insight into probabilistic reasoning because of its elegant properties. Shannon proved and built his definition of entropy on a set of simple axioms [3]:

1. $H(p(x)) \geq 0 \quad \forall x \in \mathcal{X}$.
2. $H(p(x_1), p(x_2), \dots, p(x_N))$ is a continuous and symmetric function of its arguments.
3. $H(\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N})$ is a monotonically increasing function of N .
4. $H(p(x_1), p(x_2)) = H(p(x_1)) + H(p(x_2))$ for independent events x_1 and x_2 .

5. $H(p(x_1), p(x_2), \dots, p(x_N)) = H(p(x_1)+p(x_2), p(x_3), \dots, p(x_N)) + (p(x_1)+p(x_2))H(\frac{p(x_1)}{(p(x_1)+p(x_2))}, \frac{p(x_2)}{(p(x_1)+p(x_2))})$, entropy is recursive.

Shannon proved that Equation (2.2) was the only solution for $H(X)$ that satisfied the above axioms. It is important to note that $H(X)$ is upper bounded by the Hartley information, $H(X) \leq \log |\mathcal{X}|$. As it turns out, entropy is a concave function [4] which makes it susceptible to a large area of mathematics.

2.1.1 Other Information Measurements

Entropy is only concerned with describing the uncertainty of a single source of information. However, communication systems have inputs and outputs, just like a learning machine. To characterize the relationship between two different information sources, Shannon's theory led to two well-known descriptors called the Kullback-Leibler (KL) divergence and mutual information. Consider the discrete random variable X with a finite set of possible outcomes $\mathcal{X} = \{x_k\}_{k=1}^N$. Let $p(x)$ and $q(x)$ be two probability mass functions defined on \mathcal{X} . Then the KL divergence is defined as

$$D_{KL}(p||q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} = E_p \left[\log \frac{p(X)}{q(Y)} \right]. \quad (2.3)$$

The KL divergence is a way to measure how close the distribution q is to p . When $p = q$ the KL divergence is zero. However, $D_{KL}(p||q) \neq D_{KL}(q||p)$, so it is not a true measure of distance.

Now let Y be a random variable with a finite set of possible outcomes $\mathcal{Y} = \{y_k\}_{k=1}^N$. Let the joint probability mass function between X and Y be $p(x, y)$ and the marginal probability mass functions be $p(x)$ and $p(y)$. Then the mutual information between X and Y is

$$I(X; Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} = E_{p(x, y)} \left[\log \frac{p(X, Y)}{p(X)p(Y)} \right]. \quad (2.4)$$

The mutual information measures the amount of information that one random variable contains about another. Mutual information can also be written

in terms of the KL divergence as

$$I(X; Y) = D_{KL}(p(x, y) || p(x)p(y)). \quad (2.5)$$

Thus, it is also possible to think of mutual information as measuring how close X and Y are to being independent.

2.2 Rényi Entropy

In 1960 Alfréd Rényi was searching for a general definition of information theoretic measures that would preserve the additivity property for independent events. He started with the probabilistic view of Hartley's information in Equation (2.1) since it was already shown by Shannon that this was required to preserve the additivity property. He recognized that the total amount of information developed by Shannon,

$$H(X) = \sum_{x \in \mathcal{X}} p(x) I(x), \quad (2.6)$$

assumed that the average was linear. Instead of assuming a linear relationship, he decided to apply the general theory of means. For any monotonic and continuous function $g(x)$, the general form of the mean applied to Equation (2.6) is

$$H(X) = g^{-1} \left[\sum_{x \in \mathcal{X}} p(x) g(I(x)) \right]. \quad (2.7)$$

Rényi showed that only two possible functions satisfied Equation (2.7) with the additive property [6]. One possible function was $g(x) = cx$, which is just a linear function and reduces Equation (2.7) to Shannon entropy. The other function was $g(x) = c^{-2(1-\alpha)x}$, which when applied to Equation (2.7) implies

$$H_\alpha(X) = \frac{1}{1-\alpha} \log \sum_{x \in \mathcal{X}} p^\alpha(x) \quad (2.8)$$

where $\alpha \neq 1$ and $\alpha \geq 0$. Equation (2.8) is a parametric family of functions that is referred to as the α th-order Rényi entropy.

Of special interest is the case when $\alpha = 1$. By taking the limit and applying

L'Hôpital's rule, we see that

$$\lim_{\alpha \rightarrow 1} H_\alpha(X) = \lim_{\alpha \rightarrow 1} \frac{1}{1 - \alpha} \log \sum_{x \in \mathcal{X}} p^\alpha(x) \quad (2.9)$$

$$= \frac{\lim_{\alpha \rightarrow 1} \frac{\sum_{x \in \mathcal{X}} p^\alpha(x) \log p(x)}{\sum_{x \in \mathcal{X}} p^{\alpha-1}(x)}}{\lim_{\alpha \rightarrow 1} -1} \quad (2.10)$$

$$= - \sum_{x \in \mathcal{X}} p(x) \log p(x). \quad (2.11)$$

Shannon entropy can be determined by Rényi entropy when α is near 1. Thus, Shannon entropy is just a member of the class of Rényi entropies. For this reason, Rényi entropy is often referred to as a generalized form of Shannon entropy.

2.2.1 Information Measurements of Degree α

Following Shannon's theory, Rényi also developed measurements to describe the relationship between two information sources. Consider the discrete random variable X with a finite set of possible outcomes $\mathcal{X} = \{x_k\}_{k=1}^N$. Let $p(x)$ and $q(x)$ be two probability mass functions defined on \mathcal{X} . Then the Rényi α -divergence is defined as [6]

$$D_\alpha(p||q) = \frac{1}{\alpha - 1} \log \sum_{x \in \mathcal{X}} p(x) \left(\frac{p(x)}{q(x)} \right)^{\alpha-1}. \quad (2.12)$$

By taking the limit as α approaches one and applying L'Hôpital's rule, it is easy to see that Equation (2.12) approaches the KL divergence in Equation (2.3):

$$\lim_{\alpha \rightarrow 1} D_\alpha(p||q) = \lim_{\alpha \rightarrow 1} \frac{1}{\alpha - 1} \log \sum_{x \in \mathcal{X}} p(x) \left(\frac{p(x)}{q(x)} \right)^{\alpha-1} \quad (2.13)$$

$$= \frac{- \lim_{\alpha \rightarrow 1} \sum_{x \in \mathcal{X}} p(x) \left(\frac{p(x)}{q(x)} \right)^{\alpha-1} \log \frac{q(x)}{p(x)}}{\lim_{\alpha \rightarrow 1} \sum_{x \in \mathcal{X}} \frac{p(x)}{q(x)}^{\alpha-1}} \quad (2.14)$$

$$= - \sum_{x \in \mathcal{X}} p(x) \log \frac{q(x)}{p(x)} \quad (2.15)$$

$$= \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} = D_{KL}(p||q). \quad (2.16)$$

Now let Y be a random variable with a finite set of possible outcomes $\mathcal{Y} = \{y_k\}_{k=1}^N$. Let the joint probability mass function between X and Y be $p(x, y)$ and the marginal probability mass functions be $p(x)$ and $p(y)$. Then the α mutual information between X and Y is

$$I_\alpha(X; Y) = \frac{1}{1 - \alpha} \log \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \frac{p^\alpha(x, y)}{(p(x)p(y))^{\alpha-1}} \quad (2.17)$$

$$= D_\alpha(p(x, y)||p(x)p(y)). \quad (2.18)$$

As α approaches 1, Equation (2.18) reduces to Shannon's mutual information since

$$\lim_{\alpha \rightarrow 1} D_\alpha(p(x, y)||p(x)p(y)) = D_{KL}(p(x, y)||p(x, y)). \quad (2.19)$$

2.3 Rényi Entropy vs. Shannon Entropy

The noticeable difference between Shannon entropy and Rényi entropy is the addition of the α parameter. The parametric style of Rényi entropy allows for different measurements of the uncertainty for a given distribution. When $\alpha = 0$, Equation (2.8) reduces to

$$H_0(X) = \log N = \log |\mathcal{X}|, \quad (2.20)$$

which is the Hartley information of X . As α is increased, the entropy measure places more emphasis on the events with higher probabilities. For example, when α approaches infinity, Equation (2.8) reduces to

$$H_\infty(X) = -\log \sup_{x \in \mathcal{X}} p(x). \quad (2.21)$$

The special case when $\alpha = 1$ raises important questions about the properties of Rényi entropy. Several versions of axiomatic explanations exist [7]. However, for our purpose, the most convenient set of axioms follows the same form as Shannon entropy [8]:

1. $H_\alpha(p(x)) \geq 0 \quad \forall x \in \mathcal{X}$.

2. $H_\alpha(p(x_1), p(x_2), \dots, p(x_N))$ is a continuous and symmetric function of its arguments.
3. $H_\alpha(\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N})$ is a monotonically increasing function of N .
4. $H_\alpha(p(x_1), p(x_2)) = H_\alpha(p(x_1)) + H_\alpha(p(x_2))$ for independent events x_1 and x_2 .
5. $H_\alpha(p(x_1), p(x_2), \dots, p(x_N)) = H_\alpha(p(x_1)+p(x_2), p(x_3), \dots, p(x_N)) + (p(x_1)+p(x_2))^\alpha H_\alpha(\frac{p(x_1)}{(p(x_1)+p(x_2))}, \frac{p(x_2)}{(p(x_1)+p(x_2))})$, entropy is recursive.

The main axiomatic difference between Rényi and Shannon entropy is the recursivity property. Rényi is a general form of Shannon, so the recursivity property is unique to Shannon and is one of the properties that separates it from other entropy measurements, such as Tsallis [9]. The similarity in the axiomatic framework for Rényi and Shannon implies that it may be possible to exploit the α parameter to estimate Shannon entropy. In fact, we will show that this is true when $\alpha = 2$.

2.4 Quadratic Rényi Entropy

Rényi information theoretic measures of the α th-order will be the primary interest of the rest of this thesis, specifically when $\alpha = 2$. Under this condition, Equation (2.8) reduces to

$$H_2(x) = -\log \sum_{x \in \mathcal{X}} p^2(x), \quad (2.22)$$

which we will refer to as the quadratic Rényi entropy. The first step in understanding the importance of Equation (2.22) is to compare it to Shannon's entropy. As it turns out, quadratic Rényi entropy is a lower bound of Shannon's entropy, since

$$H_1(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) \quad (2.23)$$

$$\geq -\log \sum_{x \in \mathcal{X}} p^2(x) \quad (2.24)$$

$$= H_2(X), \quad (2.25)$$

where Equation (2.24) follows by Jensen's inequality. So the quadratic entropy is an uncertainty measurement that also provides some insight about the Shannon entropy of the data.

While the relationship between Rényi's quadratic entropy and Shannon's entropy is beneficial, the true beauty of $H_2(X)$ lies in its estimator. Since the log function in Equation (2.22) is monotonically increasing, the quadratic entropy is primarily determined by the argument $\sum_{x \in \mathcal{X}} p^2(x)$. We can think of the argument as the moment of the probability mass function $E[p(x)]$. Generally, when estimating entropy, a parametric or nonparametric model is applied to the samples to calculate the PMF. However, here we are only concerned with estimating $E[p(x)]$, which is a scalar.

2.4.1 Estimating Quadratic Rényi Entropy

Since we are interested in computing the scalar value $E[p(x)]$, we have to start with the estimate for $p(x)$. Assume that a data set $\mathcal{X} = \{x_k\}_{k=1}^N$ is generated from the probability distribution $p(x)$. Then $p(x)$ can be estimated by placing some window function $W(x)$ over every sample and summing with proper normalization. This is called the Parzen-window approach and the estimator is given by [10]

$$\hat{p}(x) = \frac{1}{N\sigma} \sum_{k=1}^N W\left(\frac{x - x_k}{\sigma}\right) \quad (2.26)$$

where σ is the normalization factor. The window function can be of any form, as long as it satisfies the following properties [10]:

1. $\sup_{x \in \mathcal{X}} |W(x)| < \infty$
2. $\sum_{x \in \mathcal{X}} |W(x)| < \infty$
3. $\lim_{x \rightarrow \infty} |xW(x)| = 0$
4. $W(x) \geq 0$
5. $\sum_{x \in \mathcal{X}} W(x) = 1$

While there are numerous window functions that satisfy these properties, the Gaussian function is the best for reasons that will become apparent when we

derive the estimator. The window function will be defined as

$$W_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}}. \quad (2.27)$$

Now that we have a method for estimating $p(x)$, we can substitute this into the argument of Equation (2.22) to produce

$$\hat{H}_2(X) = -\log \sum_{x \in \mathcal{X}} \left(\frac{1}{N} \sum_{i=1}^N W_\sigma(x - x_i) \right)^2 \quad (2.28)$$

$$= -\log \sum_{x \in \mathcal{X}} \left(\frac{1}{N} \sum_{i=1}^N \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-x_i)^2}{2\sigma^2}} \right)^2 \quad (2.29)$$

$$= -\log \frac{1}{N^2} \sum_{x \in \mathcal{X}} \left(\sum_{i=1}^N \sum_{j=1}^N \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-x_i)^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-x_j)^2}{2\sigma^2}} \right) \quad (2.30)$$

$$= -\log \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \left(\sum_{x \in \mathcal{X}} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-x_i)^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-x_j)^2}{2\sigma^2}} \right) \quad (2.31)$$

$$= -\log \left(\frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \frac{1}{\sqrt{4\pi\sigma^2}} e^{\frac{-(x_j-x_i)^2}{4\sigma^2}} \right) \quad (2.32)$$

$$= -\log \left(\frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N W_{\sigma\sqrt{2}}(x_j - x_i) \right) \quad (2.33)$$

The importance of using a Gaussian becomes apparent from step (2.30) to (2.31). It was never necessary to compute the sum over the entire range \mathcal{X} because the sum of the product of Gaussians is simply the difference of their arguments and the sum of their variances. Because the log is a monotonically increasing function, we are only concerned with the argument. From now on we will refer to the argument as the Quadratic Rényi entropy estimator [11] and define it as

$$I_R(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N W_{\sigma\sqrt{2}}(x_j - x_i). \quad (2.34)$$

The estimator depends on a double summation so the algorithm is $O(N^2)$. The user also has control over the spread of the estimator with the σ parameter.

2.4.2 Quadratic Rényi Estimator vs. Shannon Entropy Estimator

As we will show now, the Quadratic Rényi estimator provides a more efficient estimator for the uncertainty of our data. For Shannon entropy we are interested in computing the scalar value $E[\log p(x)]$. We have to start with the estimate for $p(x)$. Assume that a data set $\mathcal{X} = \{x_k\}_{k=1}^N$ is generated from the probability distribution $p(x)$. Then $p(x)$ can be estimated by Equation (2.26). Using a Gaussian window and applying $\hat{p}(x)$ to Equation (2.2) results in

$$H_S(X) = - \sum_{x \in \mathcal{X}} \frac{1}{N} \sum_{k=1}^N W_\sigma(x - x_k) \log \left(\frac{1}{N} \sum_{i=1}^N W_\sigma(x - x_i) \right) \quad (2.35)$$

The log function prevents the estimator from simplifying, so the algorithm is $O(N^2|\mathcal{X}|)$.

2.5 Cauchy-Schwarz Divergence

Rényi entropy is not the only measure that reduces nicely when α is two. In fact, Rényi divergence simplifies to a well-known measurement called the Cauchy-Schwarz (CS) divergence. Lutwak [12] proved that Rényi α -divergence can be redefined as

$$D_\alpha(p||q) = \log \frac{(\sum_{\mathcal{X}} q^{\alpha-1}(x)p(x))^{\frac{1}{1-\alpha}} (\sum_{x \in \mathcal{X}} q(x)^\alpha)^{\frac{1}{\alpha}}}{(\sum_{x \in \mathcal{X}} p^\alpha(x))^{\frac{1}{\alpha(1-\alpha)}}}. \quad (2.36)$$

Just like before, as $\alpha \rightarrow 1$ Equation (2.36) reduces to $D_{KL}(p||q)$ [12]. However, now when $\alpha = 2$, Equation (2.36) will reduce to

$$D_{CS}(p||q) = -\log \frac{\sum_{x \in \mathcal{X}} p(x)q(x)}{\sqrt{\sum_{x \in \mathcal{X}} p^2(x) \sum_{x \in \mathcal{X}} q^2(x)}}. \quad (2.37)$$

This expression is referred to as the CS divergence because of its close relationship to the CS inequality

$$\sqrt{\sum_{x \in \mathcal{X}} p^2(x) \sum_{x \in \mathcal{X}} q^2(x)} \geq \sum_{x \in \mathcal{X}} p(x)q(x). \quad (2.38)$$

Using CS divergence we can also define CS mutual information. Let X and Y be random variables with joint probability distribution $p(x, y)$ and marginal distributions $p(x)$ and $p(y)$. The CS mutual information is simply

$$I_{CS}(X, Y) = D_{CS}(p(x, y) || p(x)p(y)). \quad (2.39)$$

2.5.1 Cauchy-Schwarz Divergence Estimator

Assume $\{x_i\}_{i=1}^{N_1}$ are samples generated from $p(x)$ and $\{x_i\}_{i=1}^{N_2}$ are generated from $q(x)$. By applying the same Parzen-window technique as we did before, and ignoring the log because it is a monotonically increasing function, the estimator reduces to

$$I_{CS}(p, q) = -\log \frac{\frac{1}{N_1 N_2} \sum_{i,j=1}^{N_1, N_2} W_\sigma(x_i, x_j)}{\sqrt{\frac{1}{N_1^2} \sum_{i,i'=1}^{N_1 N_2} W_\sigma(x_i, x_{i'}) \frac{1}{N_2^2} \sum_{j,j'=1}^{N_1 N_2} W_\sigma(x_j, x_{j'})}}. \quad (2.40)$$

Just like the Rényi estimator, the complexity of the algorithm is $O(N^2)$.

CHAPTER 3

REPRODUCING KERNEL HILBERT SPACES

Before continuing further with the α th-order information theoretic measures discussed in Chapter 2, we introduce a branch of mathematics that will provide a deeper geometric understanding of these measures and create a framework for a useful tool in machine learning. This branch of mathematics is called a reproducing kernel Hilbert space (RKHS) and was introduced in 1943 by Aronszajn [13]. The fundamental result of the theory is that there exists a function, called a kernel, that maps an input space to some inner product space. The advantage of this result is that a linear system that operates in the inner product space, depending on the kernel, may become a non-linear system in the input space. Also, if the RKHS dimension is high it may be easier or more efficient to compute the calculation in the input space with the kernel function. In this chapter we provide some basic definitions and theorems underlying an RKHS, and show the relationship between RKHS and estimators from Chapter 2.

3.1 RKHS Definitions

Recall that a Hilbert space is a complete vector space that is equipped with an inner product. Let \mathcal{H} be a Hilbert space defined on a domain X with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. A reproducing kernel Hilbert space is a Hilbert space \mathcal{H} associated with some kernel κ that will reproduce all functions f in \mathcal{H} by an inner product [14]. This means that for every $x \in X$ there exists a $\kappa(x, \cdot) \in \mathcal{H}$ such that

$$f(x) = \langle f, \kappa(x, \cdot) \rangle_{\mathcal{H}} \quad \forall f \in \mathcal{H}. \quad (3.1)$$

From Equation (3.1) we see that every $x \in X$ is mapped onto a function f in the RKHS by the kernel. So, the basis for the RKHS is defined by the kernel.

The function κ is called a reproducing kernel because it has the property that $\kappa(x, y) = \langle \kappa(x, \cdot), \kappa(y, \cdot) \rangle_{\mathcal{H}}$, which is evident from Equation (3.1). Not all functions are reproducing kernels, as is stated in the fundamental Theorem 3.1.1 [13].

Theorem 3.1.1 (Moore-Aronszajn theorem) *Given any positive-definite function κ , there exists a unique Hilbert space \mathcal{H} of functions for which κ is a reproducing kernel.*

A function κ is positive definite if for $\{x_k\}_{k=1}^N \in X$ and $\{c_k\}_{k=1}^N \in \mathbb{R}$

$$\sum_{i=1}^N \sum_{j=1}^N c_i c_j \kappa(x_i, x_j) \geq 0. \quad (3.2)$$

It is important to note that nothing has been said about the domain X . In fact, Theorem 3.1.1 will hold on any domain where it is possible to define a positive-definite function [15].

So far we have developed the idea that there is a relationship between some input space X and some Hilbert space \mathcal{H} when we have a positive definite function, but we have not specified what that relationship is or proved that it exists. Let X be in some n -dimensional space and let \mathcal{H} be a much higher dimensional space. Let us assume ϕ to be some kind of mapping such that $\phi: X \rightarrow \mathcal{H}$. Then, by Mercer's theorem, we are guaranteed that such a ϕ exists [16].

Theorem 3.1.1 (Mercer's theorem) *Given any positive definite function κ , we can expand this function into its eigenfunctions*

$$\kappa(x, y) = \sum_{j=1}^{\infty} \lambda_j \phi_j(x) \phi_j(y).$$

Any kernel function that satisfies Theorem 3.1.1 is called a Mercer kernel. The most common Mercer kernel is the radial basis function, or the Gaussian function

$$\kappa(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\|x-y\|^2}{2\sigma^2}}. \quad (3.3)$$

Mercer's theorem essentially provides a coordinate basis for the RKHS. Working backwards with Theorem 3.1.1, we can first construct the RKHS as a linear combination of eigenfunctions. That is, we define $\mathcal{H} = \{\sum_{j=1}^{\infty} \alpha_n \phi(x)\}$. Using these eigenfunctions we can construct a kernel. Now we can redefine

the kernel function in terms of the basis functions and provide a functional relationship between the input space X and some Hilbert space \mathcal{H} .

Definition 3.1.1 *A kernel is a function κ such that $\forall x_1, x_2 \in X$*

$$\kappa(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle_{\mathcal{H}},$$

where ϕ is a mapping such that $\phi: X \rightarrow \mathcal{H}$.

This definition of a kernel provides a new insight into the geometry of an RKHS. We can think of the kernel function as a mapping of observations from X onto an inner product space \mathcal{H} by some ϕ without ever having to compute the mapping explicitly. If we are interested in calculating the dot product of a high-dimensional vector in \mathcal{H} , we can work backwards from Mercer's theorem to compute a kernel that operates in a lower dimension while avoiding ϕ [17]. For example, the Gaussian kernel in Equation (3.3) will map the x and y vectors into an infinite-dimensional inner product space because the Gaussian has an infinite number of eigenfunctions. Definition 3.1.1 becomes especially important in cases when ϕ is nonlinear because it can be used to make nonlinear generalizations about operations in X by inner products in \mathcal{H} [18]. This is often referred to as the “kernel trick” and, as we will soon see, has a wide variety of applications in pattern analysis [19].

3.2 Density Estimation and RKHS

Recall the requirements for a valid Parzen-window when estimating a probability distribution from Chapter 2. If we also force the function W to be positive definite then the window becomes a valid kernel function. Thus, there is a close connection between density estimation and RKHS [11, 20]. Now recall the estimator for Quadratic Rényi entropy from Chapter 2

$$I_p(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N W_{\sigma\sqrt{2}}(x_j - x_i), \quad (3.4)$$

where $W_{\sigma\sqrt{2}}$ is the Gaussian function. Since $W_{\sigma\sqrt{2}}$ is a Gaussian function, it is a valid kernel function that maps x_j and x_i to some inner product space.

So we can simplify Equation (3.4) to

$$I_p(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa(x_j, x_i), \quad (3.5)$$

By Mercer's theorem we can assume that some map ϕ exists between X and \mathcal{H} to express this in terms of an inner product

$$I_p(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa(x_i, x_j) \quad (3.6)$$

$$= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}} \quad (3.7)$$

$$= \left\langle \frac{1}{N} \sum_{j=1}^N \phi(x_j), \frac{1}{N} \sum_{j=1}^N \phi(x_j) \right\rangle_{\mathcal{H}} \quad (3.8)$$

$$= \left\| \frac{1}{N} \sum_{j=1}^N \phi(x_j) \right\|_{\mathcal{H}}^2. \quad (3.9)$$

In the inner product space produced by the kernel function, the estimator for Rényi entropy is just the squared norm of the mean of the transformed data.

We can apply these same techniques to the CS divergence estimator derived in Chapter 2. Recall that the estimator is defined as

$$I_{CS}(p, q) = -\log \frac{\frac{1}{N_1 N_2} \sum_{i,j=1}^{N_1, N_2} W_{\sigma}(x_i, x_j)}{\sqrt{\frac{1}{N_1^2} \sum_{i,i'=1}^{N_1, N_2} W_{\sigma}(x_i, x_{i'}) \frac{1}{N_2^2} \sum_{j,j'=1}^{N_1, N_2} W_{\sigma}(x_j, x_{j'})}}. \quad (3.10)$$

Substituting the kernel's inner product representation and defining the mean vectors

$$\mathbf{m} = \frac{1}{N_1} \sum_{i=1}^{N_1} \phi(\mathbf{x}_i) \quad (3.11)$$

$$\mathbf{n} = \frac{1}{N_2} \sum_{j=1}^{N_2} \phi(\mathbf{x}_j), \quad (3.12)$$

it is easy to see that Equation (3.10) reduces to

$$I_{CS}(p, q) = \frac{\langle \mathbf{m}, \mathbf{n} \rangle}{\sqrt{\langle \mathbf{m}, \mathbf{m} \rangle \langle \mathbf{n}, \mathbf{n} \rangle}} = \cos \angle(\mathbf{m}, \mathbf{n}). \quad (3.13)$$

In the inner product space produced by the kernel function, the estimator for CS divergence is the angle between the mean vectors of the transformed data.

CHAPTER 4

SUPPORT VECTOR MACHINES

Up to now we have introduced this idea of α th-order information theoretic measures and shown that when $\alpha = 2$ these measurements reduce to useful estimators that can be used to give us statistical properties about the underlying data. Using this idea of reproducing kernel Hilbert spaces, we were able to show that these estimators have simple corresponding representations in some inner product space. Our goal now is to tie these measurements back into machine learning and show how they can be useful in designing a classifier. In this section we introduce the Support Vector Machine (SVM). An SVM is a supervised learning method that attempts to separate its input into a set of classes by a hyperplane. The hyperplane is chosen so that the distance from the hyperplane to the nearest class points is maximized. The SVM was built on the fundamental idea of generalization performance [21]. Generalization refers to how well a learning machine performs on test data. In this chapter we introduce some fundamental concepts in statistical learning theory that help understand and control generalization performance, describe the SVM, and propose a method that takes advantage of the discoveries in Chapter 2 and Chapter 3 to aid in designing a SVM.

4.1 Vapnik-Chervonenkis Theory

The Vapnik-Chervonenkis (VC) theory is a form of statistical learning theory that is concerned with characterizing generalization performance in learning machines. The goal is to find the best balance between the accuracy of the learning machine on a training set and its capacity. Here we define capacity as the cardinality of the largest set of points that the machine can linearly separate. To measure this capacity, Vapnik and Chervonenkis created the concept of VC dimension [22]. First we will start with the definition of

shattering. Let f be a function, or learning machine, that takes in an input and transforms it into a binary class $\{-1, 1\}$ using some kind of weights α . Assume those weights are calculated according to some training points (x_i, y_i) .

Definition 4.1.1 *A function f_α is said to shatter a set of points $\{x_1, x_2, \dots, x_n\}$ if for every possible training set $\{x_k, y_k\}_{k=1}^n$, there exists some value of α that achieves no error on the training data.*

With this formulation for shattering, we can define the VC dimension.

Definition 4.1.1 *The VC dimension is the maximum size of a set that can be shattered by the function f_α .*

The importance of VC dimension becomes important in estimating the generalization error. We can define the probability of misclassification as

$$R(\alpha) = E\left[\frac{1}{2}|y - f_\alpha(x)|\right]. \quad (4.1)$$

Assuming we are training over a data set of size N , we can calculate the error for the training set as

$$R_T(\alpha) = \frac{1}{N} \sum_{k=1}^N \left[\frac{1}{2}|y - f_\alpha(x)|\right]. \quad (4.2)$$

Vapnik [23] showed that with probability $1 - \epsilon$

$$R(\alpha) \leq R_T(\alpha) + \sqrt{\frac{d \log\left(\frac{2N}{d} + 1\right) - \log \frac{\epsilon}{4}}{N}}. \quad (4.3)$$

It was this framework that led Vapnik to develop the support vector method for calculating optimal separable hyperplanes [24].

4.2 Support Vector Machine

The maximal margin classifier was the first SVM proposed. It only works on data that is linearly separable, so the uses are rather impractical. However, it is the basis for more complex SVMs, so we introduce it here. For simplicity we will stick to a two-dimensional data set and assume binary classification, but

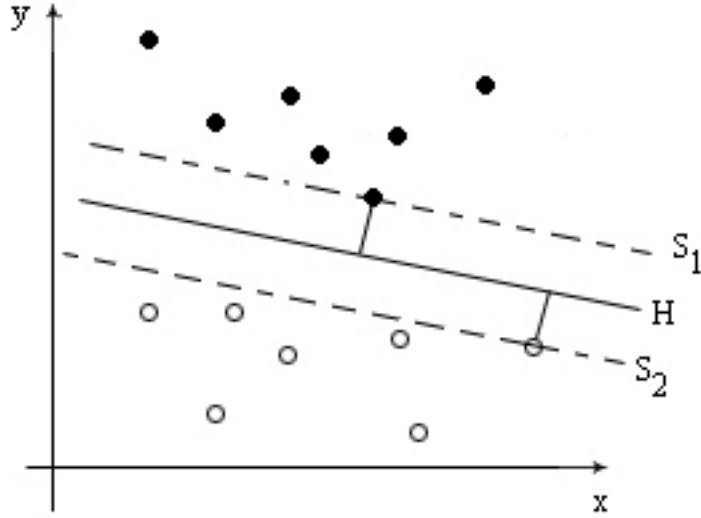


Figure 4.1: An example of a maximum-margin function H separating a training set.

the theory can be extended to n -dimensions and m -classifications. Assume we are given a set of training points $\{\mathbf{x}_k, y_k\}_{k=1}^n$ where $\mathbf{x} \in \mathbb{R}^p$ and $y \in \{-1, 1\}$. Our goal is to find a linear function H that divides the points in such a way that the line is as far away as possible from the closest sample in each set. We call these sample points support vectors. Since H is a linear function we can write it as

$$H(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b \quad (4.4)$$

where \mathbf{w} is a vector of weights and b is some offset. In order to maximize the margin between the function H and the closest sample points in each set we must find \mathbf{w} and b . Figure 4.1 shows an example of the function H that we seek. Since we are trying to maximize the margin between H and the support vectors we can define the support functions S_1 and S_2 as

$$S_1 = \langle \mathbf{w}, \mathbf{x} \rangle + b = 1 \quad (4.5)$$

$$S_2 = \langle \mathbf{w}, \mathbf{x} \rangle + b = -1. \quad (4.6)$$

Now we want to maximize the distance between S_1 and S_2 . Applying simple geometry we can see that the distance between these two functions, or hyper-

planes, is also the distance where the hyperplane intersects the line through the origin and parallel to \mathbf{w} . Let \mathbf{x}^+ correspond to the point on Equation (4.5) and \mathbf{x}^- correspond to the point on Equation (4.6). Then

$$\mathbf{x}^+ = \frac{1}{\|\mathbf{w}\|^2} \mathbf{w} \quad (4.7)$$

$$\mathbf{x}^- = \frac{-1}{\|\mathbf{w}\|^2} \mathbf{w}, \quad (4.8)$$

and the distance is

$$d(\mathbf{x}^+, \mathbf{x}^-) = \|\mathbf{x}^+ - \mathbf{x}^-\| = \frac{2}{\|\mathbf{w}\|}. \quad (4.9)$$

To maximize the distance in Equation (4.9) we need to minimize $\|\mathbf{w}\|$. However, we need to minimize $\|\mathbf{w}\|$ while ensuring our training data is appropriately classified. This results in the following optimization problem [25]

$$\min \|\mathbf{w}\| \quad \text{subject to} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad \forall i, \quad (4.10)$$

which can be rewritten in terms of Lagrange multipliers as

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle - \sum_{i=1}^n \alpha_i (y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1). \quad (4.11)$$

This optimization problem is well known and can be solved using standard quadratic programming techniques [25].

The maximal margin classifier is a good foundation for SVM theory, but since it requires the data to be linearly separable, it is not practical in real-world applications. A slack variable η was introduced into the optimization problem to calculate the hyperplane that would separate the data as cleanly as possible. As a result, the optimization problem in Equation (4.10) becomes [25]

$$\min \|\mathbf{w} + C \sum_{i=1}^n \eta_i\| \quad \text{subject to} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \eta_i \quad \forall i. \quad (4.12)$$

4.3 Kernel Trick

Recall that the goal of the SVM is to find a linear function H that satisfies Equation (4.4) and separates the data as best as possible. The function is obtained by minimizing $\|\mathbf{w}\|$ with some constraint. Because H is a linear function, the weight vector will always be a linear combination of the training samples [25]

$$\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i. \quad (4.13)$$

Plugging this back into Equation (4.4) we get

$$H(\mathbf{x}) = \sum_{i=1}^N \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b. \quad (4.14)$$

Note the SVM operates in an inner product space. Recall from Chapter 2 that we can represent inner product spaces using a kernel function. With this in mind, we can reformulate the problem.

Let $\phi : X \rightarrow \mathcal{H}$ be some kind of mapping from the input space X to some feature space \mathcal{H} . Let us assume the mapping takes the data in X which is not linearly separable, and makes it linearly separable in \mathcal{H} . Let us apply the SVM method to this feature space \mathcal{H} . Then we are looking for some function H such that

$$H(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b. \quad (4.15)$$

But by Equation (4.13) we know \mathbf{w} is a linear combination of the training samples so

$$H(\mathbf{x}) = \sum_{i=1}^N \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle + b. \quad (4.16)$$

Applying the kernel trick we see that

$$H(\mathbf{x}) = \sum_{i=1}^N \alpha_i y_i \kappa(\mathbf{x}_i, \mathbf{x}) + b. \quad (4.17)$$

Let us examine what just happened. The feature space X contains data that is not linearly separable (Figure 4.2). However, using a kernel function, we are able to use some mapping ϕ to simultaneously represent our data in a linearly separable feature space. Applying the SVM in this feature space is

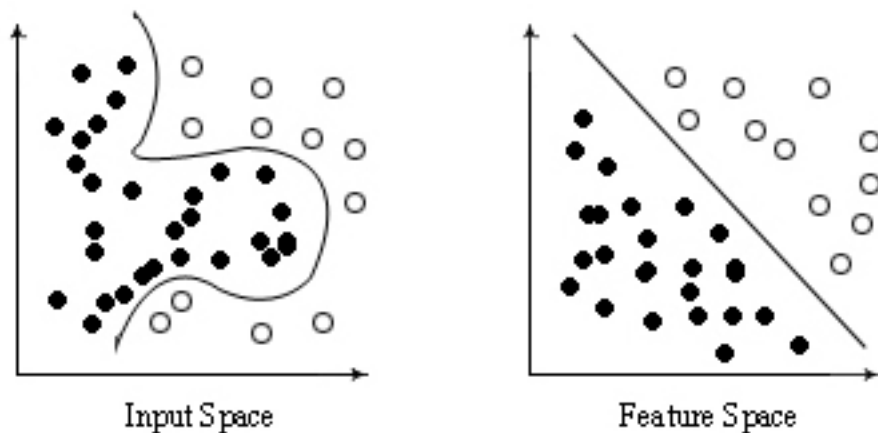


Figure 4.2: Example of the input space mapped to some feature space via a kernel function.

identical to applying some non-linear fitting model to our input space.

4.4 Evaluating Kernels Using Information Theory

The kernel method allows us to create non-linear classifiers. Of course, this is assuming the kernel is able to map the input space to a feature space where our data is linearly separable. Determining how well the data is linearized in the feature space requires us either to calculate the mapping ϕ , or to compute and test the classifier. Both of these options have significant drawbacks. In some cases it is not possible to calculate the mapping. For example, consider the Gaussian kernel case. If we apply Mercer's theorem we see that the inner product space is infinite-dimensional. In other cases, computing or testing the classifier is too time consuming. We propose a method for evaluating how well a kernel linearizes a feature space by exploiting information theoretic measurements of the training data.

Recall what happens in the feature space when we estimate the Rényi entropy and the Cauchy-Schwarz divergence from Chapter 3. In the inner product space produced by the kernel function, the estimator for Rényi en-

tropy is just the squared norm of the mean of the transformed data

$$I_p(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa(x_i, x_j) \quad (4.18)$$

$$= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}} \quad (4.19)$$

$$= \left\langle \frac{1}{N} \sum_{j=1}^N \phi(x_i), \frac{1}{N} \sum_{j=1}^N \phi(x_j) \right\rangle_{\mathcal{H}} \quad (4.20)$$

$$= \left\| \frac{1}{N} \sum_{j=1}^N \phi(x_i) \right\|_{\mathcal{H}}^2 \quad (4.21)$$

$$= \|\mathbf{m}\|_{\mathcal{H}}^2. \quad (4.22)$$

The estimator for CS divergence is the angle between the mean vectors of the two transformed data classes

$$I_{CS}(p, q) = \frac{\langle \mathbf{m}, \mathbf{n} \rangle}{\sqrt{\langle \mathbf{m}, \mathbf{m} \rangle \langle \mathbf{n}, \mathbf{n} \rangle}} = \cos \angle(\mathbf{m}, \mathbf{n}). \quad (4.23)$$

If we use these estimates on the training data, we can characterize how well the classes are separated in the feature space.

Let us assume $\{\mathbf{x}_k, y_k\}_{k=1}^n$ where $\mathbf{x} \in \mathbb{R}^2$ and $y_k \in \{-1, 1\}$. If we calculate the Quadratic Rényi entropy for each class by applying the estimator from Equation (2.34) then we will get the magnitude of the mean vector in the feature space. Let $r_1 = \|\phi_1(\mathbf{x})\|$ and $r_2 = \|\phi_{-1}(\mathbf{x})\|$ represent the magnitudes for each class. Since $\mathbf{x} \in \mathbb{R}^2$ we know the mean of each class must lie on a circle of radius r_1 and r_2 . If the mean values are close, or if $r_1 = r_2$, then all we know is that the mean of both our feature vectors lies on the radius of the same circle. But, if we use the CS divergence measure to calculate the angle between the two classes, we can figure out if the mean feature vectors have any separation on that circle. If the mean feature vectors vary enough, i.e. if $r_1 \gg r_2$, then it is possible to assume the classes are separated well enough in the feature space without having to calculate the angle between them. Estimating these information theoretic measures using the input data allows us to evaluate how well the kernel function separates the data without having to explicitly compute the mapping or build the classifier.

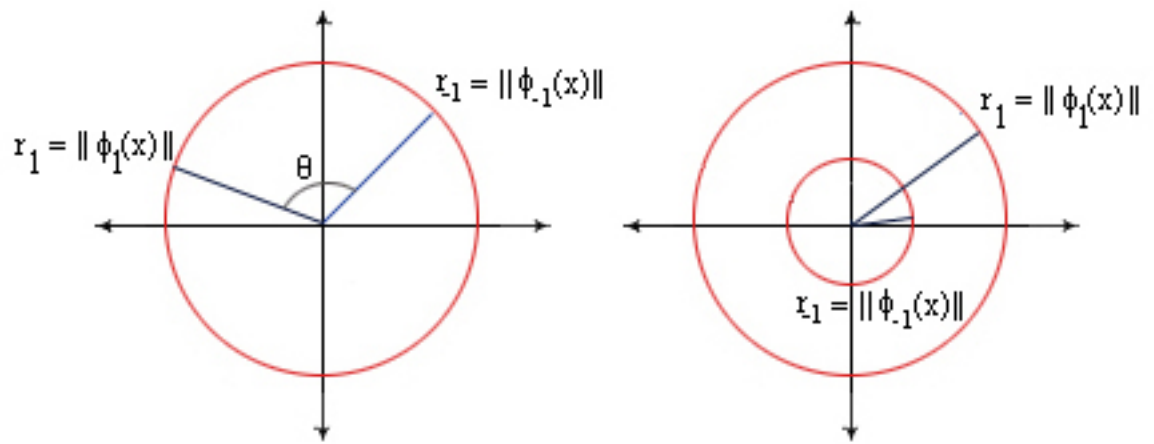


Figure 4.3: Using the Rényi entropy and CS divergence measures it is possible to classify how separated the classes are in the feature space.

CHAPTER 5

CONCLUSION

In most cases, the only way of evaluating how well a kernel operates in a support vector machine on a given set of input data is to build the classifier and test it on real data. This can take a long time depending on the size of the training set and the amount of available computational power. This thesis proposed a method for evaluating how well a kernel will separate classification classes in some feature space using information theoretic measure estimates on the input data. Even though the proposed method uses estimators that operate on the order of $O(N^2)$, this operation can be much quicker than the standard quadratic optimization technique used in the SVM classifier. Furthermore, the proposed method implies that it may be possible to build an adaptive SVM that can evaluate how well the kernel is performing and make adjustments if necessary.

This thesis focused on information theoretic measures and how they could explain the kernel operations. Future research could be dedicated to finding new estimators that provide other important metrics in the feature space, for example, an estimator for the distance between two vectors. The proposed estimators are also primarily concerned with the mean values of the data in the feature space. This raises the question of whether or not it is possible to create estimators that can classify the variance of the data in the feature space.

REFERENCES

- [1] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. San Diego, CA: Academic Press, Oct. 1990.
- [2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York, NY: Wiley-Interscience, Nov. 2001.
- [3] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, 1949.
- [4] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ: Wiley-Interscience, 2006.
- [5] R. V. L. Hartley, “Transmission of information,” *Bell Syst. Tech. Journal*, vol. 7, pp. 535–563, 1928.
- [6] A. Rényi, “On measures of entropy and information,” in *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability*, 1960, pp. 547–561.
- [7] J. Aczél and Z. Daróczy, *On Measures of Information and Their Characterizations*. New York: Academic Press [Harcourt Brace Jovanovich Publishers], 1975.
- [8] A. Rényi and P. Turán, *Selected papers of Alfred Rényi*. Budapest: Akadémiai Kiadó, 1976.
- [9] C. Tsallis, “Possible generalization of Boltzmann-Gibbs statistics,” *J. Statist. Phys.*, vol. 52, no. 1-2, pp. 479–487, 1988.
- [10] E. Parzen, “On estimation of a probability density function and mode,” *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.
- [11] J. Principe and D. Xu, “Information-theoretic learning using Rényi’s quadratic entropy,” in *Proc. 1st International Workshop on Independent Component Analysis and Signal Separation*, Jan. 1999, pp. 407–412.

- [12] E. Lutwak, D. Yang, and G. Zhang, “Cramerrao and moment-entropy inequalities for Renyi entropy and generalized Fisher information,” *IEEE Transactions on Information Theory*, vol. 51, no. 2, pp. 473–478, 2005.
- [13] N. Aronszajn, “Theory of reproducing kernels,” *Transactions of the American Mathematical Society*, vol. 68, pp. 337–404, 1950.
- [14] L. Máté, *Hilbert Space Methods in Science and Engineering*. Budapest, Akademiai Kiado: A. Hilger, 1989.
- [15] A. Berlinet and C. Thomas-Agnan, *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Norwell, MA: Kluwer Academic, 2004.
- [16] J. Mercer, “Functions of positive and negative type and their connection with the theory of integral equations,” *Philos. Trans. Royal Soc. (A)*, vol. 83, no. 559, pp. 69–70, Nov. 1909.
- [17] A. Aizerman, E. M. Braverman, and L. I. Rozoner, “Theoretical foundations of the potential function method in pattern recognition learning,” *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.
- [18] B. Schölkopf, A. Smola, and K.-R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, July 1998.
- [19] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY: Cambridge University Press, 2004.
- [20] M. Girolami, “Orthogonal series density estimation and the kernel eigenvalue problem,” *Neural Comput.*, vol. 14, no. 3, pp. 669–688, Mar. 2002.
- [21] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: <http://dx.doi.org/10.1007/BF00994018>
- [22] V. N. Vapnik and Ya, “On the uniform convergence of relative frequencies of events to their probabilities,” *Theory of Probability and its Applications*, vol. 16, no. 2, pp. 264–280, 1971. [Online]. Available: <http://link.aip.org/link/?TPR/16/264/1>
- [23] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY: Springer-Verlag, New York, Inc., 1995.
- [24] V. N. Vapnik, *Statistical Learning Theory*. New York, NY: Wiley, Sep. 1998.
- [25] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. New York, NY: Cambridge University Press, 2000.